Data Visualisation with IRIS Explorer

Jeremy Walton *NAG Ltd, Oxford, UK*

Abstract

The use of IRIS Explorer, a data visualisation toolkit, is described with reference to some examples from the fields of chemistry, computational fluid dynamics and finite element analysis. Some of the new features in the latest version of the toolkit are highlighted, and its relationship to other technologies such as the NAG numerical library, Open Inventor and VRML are discussed in detail.

Keywords: IRIS Explorer; Open Inventor; software re-use; VRML

1 Introduction

The important role which visualisation plays in the interpretation of data has been recognised for a long time. Much progress has been made in academic research and industrial development towards the goal of producing modern data display software that is powerful, and yet easy to invoke, modify and extend. In this paper we discuss the use of IRIS Explorer, a data visualisation toolkit, in the analysis of scientific data, and outline the way in which it makes use of complementary software technologies in its architecture. One of these is Open Inventor, an object-oriented 3D graphics library with an associated file format that was used as the basis of the Virtual Reality Modelling Language (VRML), which is the description language for 3D scenes on the WWW.

Our paper is arranged as follows. We introduce IRIS Explorer as a data visualisation toolkit in the following section, and follow this in §3 with a description of the software technologies that form a basis for the system. Specifically, we shall look at the NAG libraries (§3.1), Open Inventor (§3.2) and VRML (§3.3). Some of the new features that have been added in the latest release of IRIS Explorer are then discussed in §4, before concluding (§5) with some closing remarks.

2 IRIS Explorer—a visualisation toolkit

Data visualisation is the gaining of insight by making a picture out of numbers, be they from engineering, geology, physics, chemistry, finance or architecture. In the past, visualisation applications have been constructed using traditional programming methods incorporating calls to a graphics library such as—for example—the OpenGL [1]. However, the drive towards applications that are easier to construct, maintain and modify has led to the introduction of libraries such as Open Inventor [2], which provides a set of object-oriented abstractions above OpenGL, and—at a still higher level of abstraction—toolkits such as IRIS Explorer [3], which give users the ability to construct and edit visualisation applications using an intuitive visual programming metaphor.

The hierarchical relationship between these packages (IRIS Explorer uses Open Inventor to create pieces of geometry—lines, surfaces, voxels, etc.—and Open Inventor uses OpenGL to render this geometry in a graphics window) can often be exploited by application developers. For example, new geometry manipulation modules can be built in IRIS Explorer using components

from Open Inventor, and new rendering methods can be created in Open Inventor by making calls to OpenGL. We shall return to this point below, in §3.2.

When working with IRIS Explorer (or other toolkits such as AVS [4], Data Explorer [5] and Khoros [6]) users interactively create their application in the form of a network (or *map*) of *modules*. Here, each module is a software routine that operates on its input data to produce some output. The map—built by connecting module inputs and outputs—defines the way in which data flows through the application. Editing of the map is performed via a point-and-click programming interface, where modules are selected from the *module librarian*, dragged onto the *map editor* and connected together. The behaviour of each module is usually controlled by some set of *parameters* (such as the name of an input file, the value for which an isosurface is to be calculated, or the form of a colourmap) and the user can interact with these while the application is running via a standard set of widgets. Some of these elements are illustrated in the IRIS Explorer screen shot in Figure 1. A particular feature of the toolkit is that it is *extensible*; users have the facility to create their own, supplementary, modules which can be added to the default set.

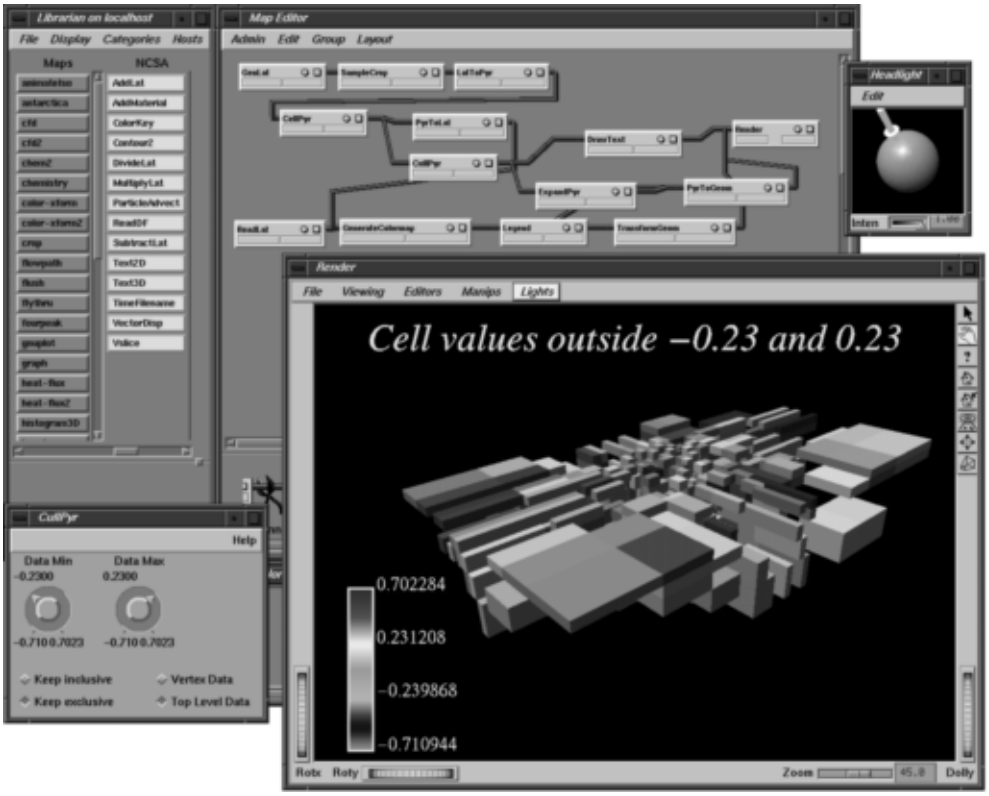

Figure 1: Running a map in IRIS Explorer. The module librarian is at top left, and the map editor is in the middle at the top. The control panel for one of the modules in the map is at bottom left. The final scene appears in the large display window of the *Render* module, together with annotation elements created by other modules in the map. This map is displaying cell-based data from an oil reservoir simulation.

IRIS Explorer was originally developed by SGI for SGI workstations, and ported by NAG to other platforms. Development was taken over by NAG in 1994, and the product is now available from NAG for SGI, Cray, Sun (SunOS and Solaris), HP 9000/700, IBM RS/6000, DEC Alpha and—most recently—Windows NT platforms (see Figure 2). In §4 below, we will outline some of the new features of Release 3.5, which is the latest version of IRIS Explorer. Before doing

that, however, we interpose a section which looks at some of the libraries that act as a foundation for IRIS Explorer, and shows how each is utilised within the modules and the base system.

3 IRIS Explorer and other software technologies

3.1 The NAG libraries

The numerical and graphics subroutine libraries from NAG are widely known and respected. In particular, the NAG numerical library, which contains more than 1000 routines, has been used by a large community of developers for over twenty years, and provides easy access to complex and highly sophisticated algorithms for performing tasks such as minimisation, solution of ordinary and partial differential equations, quadrature, statistical analysis and solving linear and non-linear equations. The advantages of using a high-quality numerical library are well-known: development time is saved, investment in application programs is protected, and the testing of the accuracy and reliability of the routines gives confidence in the solution.

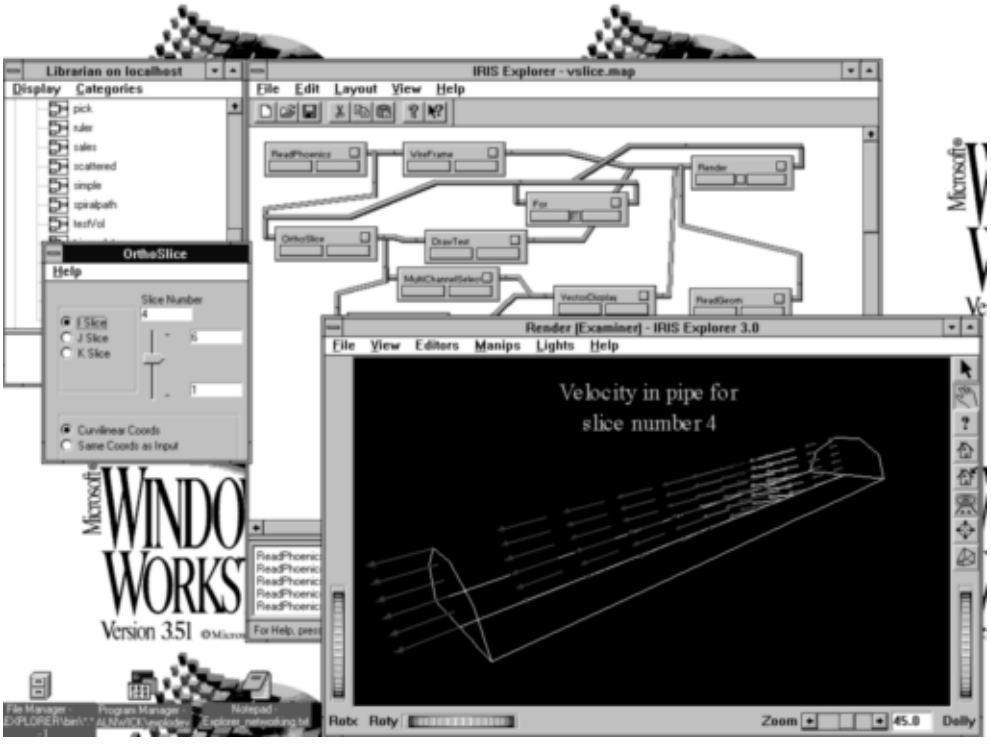

Figure 2: Displaying the flow through a half-cylindrical pipe using the Windows NT port of IRIS Explorer. The fluid velocity has been calculated using the cfd package PHOENICS, and read into the map using the *ReadPhoenics* module. The control panel for the *OrthoSlice* module is on the left, in front of the module librarian. This map features animation; the *For* module is used to step through the different slices, while the caption updates dynamically as the slice number changes.

We have made use of the numerical library in the creation of some of the modules in IRIS Explorer. An example is provided by the modules for calculating streamlines for particles released into vector fields. At the heart of this calculation is the solution of the velocity equation for the particle—a first-order ordinary differential equation—and the IRIS Explorer modules call the appropriate routine from the NAG library to obtain this. We have shown elsewhere [7,8] that

the use of this routine gives superior results for simple test cases when compared to modules from other visualisation systems that use simpler, less sophisticated algorithms.

More broadly, such work has highlighted the importance of a good understanding (or at least awareness) of the details of the algorithm inside a module in finding out (a) how it works and (b) how reliable it is. The use of a good numerical library (such as the one from NAG) in the construction of a module is the key to this, since, as noted above, it provides the user with a high degree of confidence in the knowledge that the implementation of the algorithm has been extensively tested and used in a wide range of other applications. Moreover, it provides the module with a ready-made reference for documentation purposes.

The NAG graphics library, which provides user-callable routines for numerical plotting, is built on many of the routines from the numerical library. This has been used to create IRIS Explorer modules for simple plotting of y(x) line graphs and z(x,y) contours.

Work on incorporating elements of the NAG library into IRIS Explorer is an on-going project. For example, the latest release (see §4, below) makes use of new interpolation routines from the NAG numerical library. In addition, the annotation modules have been updated to handle inline font changes, subscripts, superscripts and multiline fractions via calls to the NAG graphics library.

3.2 Open Inventor—a 3D toolkit

Open Inventor [2] is an object-oriented toolkit that allows users to create 3D graphics applications. It provides a comprehensive software model for developing interactions with the elements in the 3D scene, and defines an intuitive file format for 3D scenes. Open Inventor is written in C++, and uses OpenGL for rendering scenes.

An Open Inventor application creates and manipulates 3D scenes in a database known as a *scene graph* of objects called *nodes*. Various classes of nodes implement different geometry elements (primitive shapes, surfaces, text, etc.), properties (colour, lighting, texturing, transformation, etc.) as well as other behaviours. Other nodes help to organise the structure of the scene graph—for example, nodes may be grouped together under a separator node so that they do not affect other nodes outside the group. The ordering of the nodes in the scene graph and their relationship to one another defines the 3D scene.

Having created the scene graph, actions can then be applied to it; these include rendering, computing a bounding box or writing to a file. The file format used by Open Inventor is rather intuitive, and relates directly to the structure of the scene graph. The simplicity of this format, coupled with the fact that, once the scene has been created within an Open Inventor application, it may be easily written to a file, has led to its adoption as a de facto standard for 3D geometry files and contributed to its being used as the precursor of VRML 1.0 (see following subsection). A number of translators between Open Inventor and other formats have already appeared and it is straightforward to exploit this when passing 3D data between one Open Inventor-based application and another. In some cases, this can be performed using a familiar cut and paste mechanism, exactly analogous to the transfer of more conventional data types such as text.

Open Inventor supports the creation of new nodes both through extensions to the existing library, and through collecting nodes together into higher-level abstractions. Extensions are easy to do because the library is written in C++, and new objects can be created by subclassing from the orginal set of C++ classes [9]. Because of this, and the fact that executables can dynamically load shared objects in a modern operating system, it is straightforward to share these new nodes with other Inventor applications. The creation of higher-level abstractions of nodes is handled in

Inventor through the mechanism of *node kits* [2, 9], which define collections of nodes using a template which determines the arrangement of the nodes and their insertion points into the scene graph.

IRIS Explorer uses Open Inventor to create and manipulate geometry (shapes, lines, polygons, etc.). More technically, the IRIS Explorer geometry datatype *is* an Open Inventor scene graph. This means that the module developer in IRIS Explorer can make use of all of the functionality in Open Inventor when working with geometry. Conversely, it is often possible to take existing Open Inventor applications and turn them into modules. Finally, the common file format can be used to pass 3D scenes between IRIS Explorer and other Open Inventor applications. For example, Axiom, NAG's symbolic solver, uses Open Inventor as its 3D graphics library when rendering views of the mathematical surfaces which it manipulates, and the common basis of the two systems has been demonstrated [10] by sharing 3D objects between them.

Once again, the close links between IRIS Explorer and this software technology are being further exploited in the latest release (see §4, below) with the addition of 3D cut and paste from the ***Render*** module, improved picking and editing of scenes, and new Open Inventor-based modules. The subclassing mechanism has been exploited in this release in the creation of new geometry nodes associated with the so-called "smoke" techniques for visualising vector fields (see Figure 3). Other work involving the addition of new nodes via node kits is discussed in §5.

3.3 VRML—3D on the WWW

As noted above, the ubiquitousness of the Open Inventor file format was further enhanced when it was adopted as the basis of VRML, the language used to define the interchange of 3D geometry on the WWW. This is the analogue, in a 3D world, of HTML, which is used to define text-based documents on the Web. The use of an already extant language had the advantage that existing Open Inventor content and applications could be easily converted to VRML, and so shared with other workers on the Web. VRML 1.0 is a subset of the file format defined by Open Inventor 2.1. The definition of version 2.0 of the language has been recently finalised; below, in §5, we shall give a brief account of its features. However, much work has already been done using VRML 1.0, and a number of VRML applications and browsers (including WebSpace [11], which was developed by Silicon Graphics, and ported to a variety of other platforms [12] by Template Graphics Software) have been in use for some time. The widespread acceptance of VRML makes it a natural choice for storing and sharing 3D geometry as output from a visualisation package. In the case of IRIS Explorer, converting its output to VRML is most easily done using the Inventor translator `ivToVRML`, which is part of the WebSpace distribution.

Some of the uses of IRIS Explorer with VRML have been reviewed recently [13], including a visualisation web server [14, 15] and a number of examples using EyeChem, a purpose-built suite of IRIS Explorer modules for molecular visualisation [16-18].

4 What's new in IRIS Explorer 3.5?

IRIS Explorer 3.5 contains a number of new features and enhancements to existing modules. Internally, the system has been ported to use Open Inventor 2.1, which gives a speedup of between 20 and 400% for rendering, as well as other improvements such as full VRML support, a new locate highlighting node and a new all-in-one geometry manipulator. In addition, the ***Render*** module has been completely rewritten for this release to incorporate new features such as 3D cut and paste and improved support for picking, editing and saving elements from the scene. Software rendering via X has also been added, which means that IRIS Explorer can now run on

any display, even those that do not support OpenGL. Other internal enhancements include the incorporation of NAG Library interpolation routines into the system library. The NAG routines use a state-of-the-art algorithm, which results in faster, more accurate interpolation for all modules that make use of this facility.

Functionality that has been added at 3.5 includes new pyramid modules for analysing data on irregular grids using contouring, streamlines, displacement maps and slicing. In addition, support for datasets which are associated with elements of the grid such as edges, faces and volumes (as well as nodes) has been substantially improved. Other new modules have been added to enhance the display of multivariate data and to manipulate geometry.

This last category of extensions provides an example of how the functionality of Open Inventor can be exploited within IRIS Explorer; since IRIS Explorer's geometry type is an Open Inventor scene graph, all of the classes and methods available in that library can be invoked in geometry-based modules, and existing Open Inventor routines can be easily converted to IRIS Explorer modules. In a similar way, the new ***LatSmoke*** module illustrates how new Inventor nodes can be created by subclassing and used in novel visualisation methods. LatSmoke (originally developed at the Lawrence Livermore National Laboratory [19], and enhanced in 3.5 by the incorporation of NAG library routines) generates particle traces in a vector field using small translucent tetrahedra (see Figure 3), which is a particularly effective technique for visualising turbulent flows. The implementation is such that the smoke probe can be interactively moved around in 3D space, by means of an Inventor manipulator. Computing and rendering the flow volume is fast enough for the smoke to be drawn continuously while the probe is moving or the scene is rotating [19].

Besides adding new modules, extra functionality has been incorporated into existing modules. For example, solid contouring in 2D and 3D has been added as an option to the ***LatToGeom*** and ***IsosurfaceLat*** modules. The NAG Graphics modules which appeared at Release 3.0 have been updated to support geometry output, which means that it is now possible to construct composite displays of 1D, 2D and 3D data in the ***Render*** window. Finally, a number of enhancements have been added for annotation of the rendered scene. Thus, for example, the ***DrawText*** module (introduced at Release 3.0), now supports inline font changes, subscripting and superscripting of text and multiline fractions, in addition to existing features like embedded parameter values in the text, support for multiple lines of text, and control over colour and placement of text. In addition, the ***Annotation*** module has been rewritten to incorporate multiple labels, each with their own font, size and colour, and lines and arrows for pointing at objects in the scene. Some of the new functionality in IRIS Explorer 3.5 is illustrated in Figure 3, which shows a selection of scenes which have been created using the new and enhanced modules.

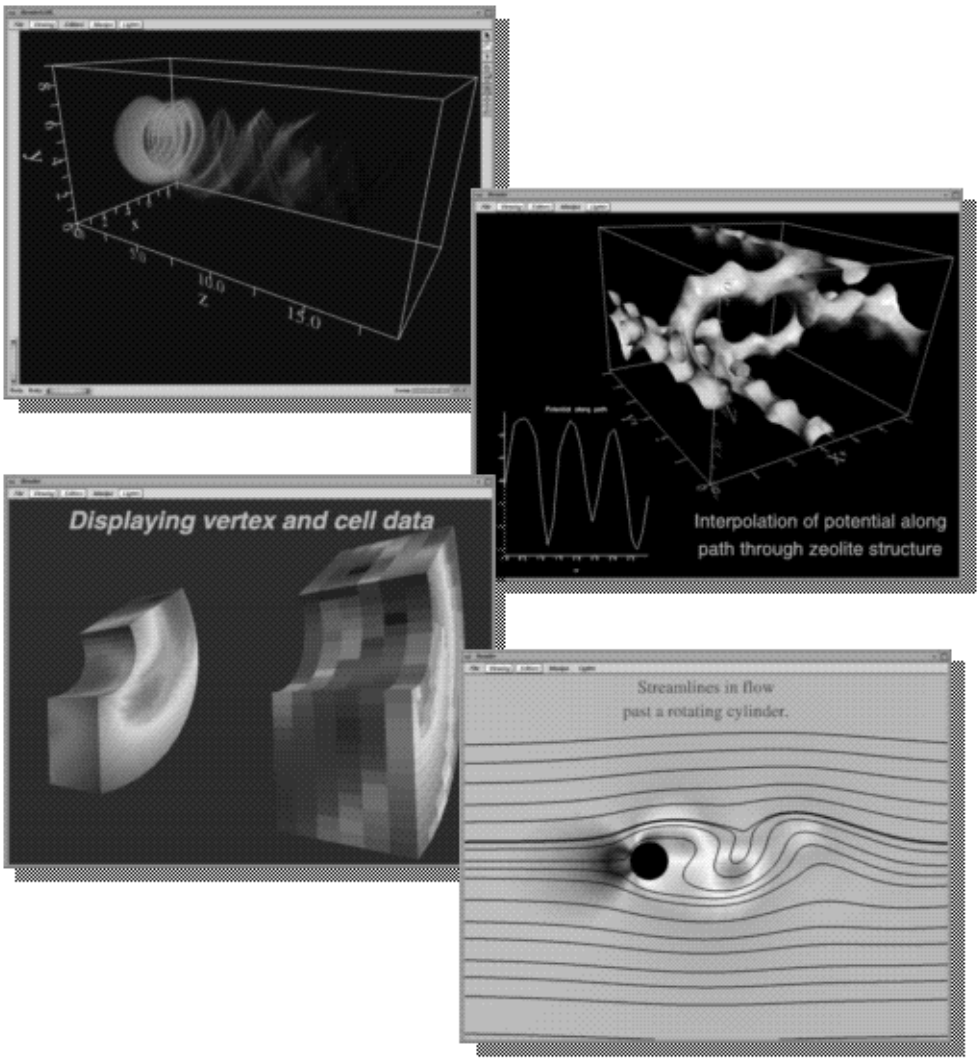

Figure 3: Some examples of scenes created using IRIS Explorer 3.5. Top left: Displaying vector data using smoke trails with the new *LatSmoke* module. Top right: Plotting the potential energy along a path through a zeolite crystal; the plot is incorporated into the display in the *Render* window, using the new version of the *NAGGraph* module. Bottom left: Displaying results from a finite element calculation on a cylinder segment, where the geometry has been coloured according to both vertex-based data (on the left) and cell-based data (on the right). Bottom right: Streamlines for flow past a rotating cylinder. The flow is calculated on an irregular grid, and the streamlines come from the new *StreamlinePyr* module.

5 Conclusions and future work

In this paper, we have introduced IRIS Explorer as a visualisation toolkit, illustrated its use with some examples, and discussed some of the software upon which it is constructed. Building a system on top of other complementary subsystems is a well-established practice in software design, and leads to efficient re-use of code and the inheritance of new features and functionality as the underlying components evolve. For example, because IRIS Explorer 3.5 uses Open Inventor 2.1, it *automatically* picks up all of the new functionality (and in particular, the performance improvements) of that release. Another example is provided by the utilisation of the NAG library. The use of up-to-date, sophisticated numerical methods is as important in the visualisation of data as in its computation [8], although their implementation can often present a

challenge to the developers of the visualisation system. Here, it makes sense to build on the work of others by incorporating calls to a reliable source of numerical routines such as that provided by NAG.

The close relationship between IRIS Explorer and Open Inventor has been discussed throughout this paper. The incorporation of Inventor-based applications and libraries into IRIS Explorer is straightforward, because they share the same underlying basis. For example, the recently released GraphMaster library from Template Graphics Software (TGS), which contains nodes for axes, labels and other components necessary for producing good-quality y(x) plots, can be easily incorporated into IRIS Explorer modules because it is implemented as a suite of Open Inventor node kits. In a similar way, TGS's VRMLMaster library, which is a set of extensions to Open Inventor 2.1 for the creation and manipulation of VRML 2.0 scenes, could also be added in for users who wish to develop modules to handle 3D scenes which incorporate new VRML 2.0 features such as behaviour, sensors and prototyping. Finally, the TGS PrintMaster product can be used with IRIS Explorer to generate high quality vector hardcopy output from the Inventor scenes created within the ***Render*** module. Users of IRIS Explorer can expect to see these enhancements (and others) appearing in the product in the near future.

References

1. Nieder, J., Davis, T. and Woo, M., OpenGL Programming Guide. The Official Guide to Learning OpenGL, Release 1, Addison Wesley (1993).
2. Wernecke, J., The Inventor Mentor. Programming Object-Oriented Graphics with Open Inventor, Release 2, Addison Wesley (1994).
3. The Numerical Algorithms Group, IRIS Explorer User's Guide (1995).
4. Upson, C., Faulhaber Jr., T., Kamins, D., Laidlaw, D., Schlegel, D., Vroom, J., Gurwitz, R. and van Dam, A., "The Application Visualisation System: Computational Environment for Scientific Visualisation", IEEE Computer Graphics and Applications, **9**, 30 (1989).
5. Lucas, B., Abram, G.D., Collins, N.S., Epstein, D.A., Gresh, D.L. and McAuliffe, K.P., "An Architecture for a Scientific Visualisation System", Proceedings of Visualisation '92, IEEE Computer Society Press, 107 (1992).
6. Rasure, J. and Young, M., "An Open Environment for Image Processing Software Development", Proceedings of 1992 SPIE/IS&T Symposium on Electronic Imaging, 1659 (1992).
7. Rushmeier, H., Botts, M., Uselton, S., Walton, J., Watkins, H. and Watson, D., "Metrics and Benchmarks for Visualization", Proceedings of Visualisation'95, IEEE Computer Society Press, 422 (1995).
8. Walton, J., "Visualisation Benchmarking: A Practical Application of 3D Publishing", Eurographics UK 1996 Conference Proceedings **2**, 339 (1996).
9. Werneke, J., The Inventor Toolmaker. Extending Open Inventor, Release 2, Addison Wesley (1994).
10. Walton, J. and Dewar, M., "See what I mean? Using Graphics Toolkits to Visualise Numerical Data", in proceedings of VisMath'95, Berlin, to appear (1996).
11. http://www.sgi.com/WebFORCE/WebSpace/
12. `http://www.sd.tgs.com/~template/WebSpace`
13 Walton, J, "Publishing in a 3D World: Data Sharing with VRML", in proceedings of BCS conference on "3D and Multimedia on the Internet, WWW and Networks", Bradford (1996).
14. Wood, J., "IRIS Explorer, VRML and the WWW", poster presentation at the IRIS Explorer User Group meeting, SIGGRAPH 95 (1995).

15. Wood, J., Brodlie, K.W. and Wright, H., “Visualisation over the World Wide Web and its Applications to Environmental Data”, Proceedings of Visualisation’96, IEEE Computer Society Press, to appear (1996).
16. http://www.ch.ic.ac.uk/VRML/
17. Casher, O. and Rzepa, H.S., “A Chemical Collaboratory using Explorer EyeChem and the Common Client Interface”, Computer Graphics **29**, No. 2, 52 (1995).
18. Casher, O., Leach, C., Page, C.S. and Rzepa, H.S., “Advanced VRML-Based Chemistry Applications: A 3D Molecular Hyperglossary”, 2nd Electronic Computational Chemistry Conference (http://www.ch.ic.ac.uk/eccc2/) (1995).
19. Max, N., Becker, B. and Crawfis, R., “Flow Volumes for Interactive Vector Field Visualisation”, Proceedings of Visualisation’93, IEEE Computer Society Press, 19 (1993).